Computer Security Symposium 2021
26 - 29 October 2021

グラフ学習による ゲートレベルハードウェアトロイ識別と評価

山下 一樹[1,a)]　長谷川 健人[2]　披田野 清良[2]　清本 晋作[2]　戸川 望[1]

概要：近年では，IC 製品の需要拡大に伴い設計や製造工程の外部委託が増加している．工程の一部に第三者が加わることでハードウェアトロイと呼ばれる悪意ある機能を持つ回路が挿入されるリスクが深刻となっている．ハードウェアトロイは情報漏洩や機能障害などの思わぬ動作を発生させることで IC 製品に甚大な被害を引き起こすため，識別手法の開発が急務である．本稿ではゲートレベルネットリストを対象に，グラフ学習を用いたハードウェアトロイ識別手法を提案し評価する．本手法では，まずゲートレベルネットリストのゲートをノード，信号線をエッジに対応させグラフを生成する．この際，各ノードにゲートの特徴を表す特徴ベクトルを与える．次にグラフの各ノードについて，近傍ノードの特徴ベクトルを考慮した新たな特徴ベクトルを生成する．特徴ベクトルの更新を繰り返すことでグラフ全体の学習を可能とする．得られた特徴ベクトルの値によって各ゲートがハードウェアトロイを構成するゲートかどうかを識別する．17 種類のゲートレベルネットリストを用いた実験により，提案手法の有効性を確認した．

キーワード：ハードウェアトロイ，グラフ学習，機械学習，ネットリスト，論理ゲート

Hardware-Trojan Detection Utilizing Graph Neural Networks at Gate-Level Netlists

Kazuki Yamashita[1,a)]　Kento Hasegawa[2]　Seira Hidano[2]　Shinsaku Kiyomoto[2]
Nozomu Togawa[1]

Abstract: Recently, the threat of injecting a circuit with malicious functions called a hardware trojan has been growing due to the increase in outsourcing of IC design and manufacturing processes. In this paper, we propose a hardware-Trojans detection method based on graph neural networks targeting gate-level netlists. First, we convert the gates and wires of a given gate-level netlist into nodes and edges, respectively, and create a graph structure in which each node is given a feature vector representing the gate features. Next, for each node in the graph, a new feature vector is generated by considering the feature vectors of neighboring nodes. By repeatedly updating the feature vectors, the entire graph can be learned and the Trojan gates can be detected. The effectiveness of the proposed method is confirmed by experimental evaluations.

Keywords: hardware Trojan, graph neaural networks, machine learning, netlist, logic gate

1. はじめに

近年，様々なモノの IoT (Internet of Things) 化が進んだことで IC の需要が高まり，ハードウェア製品の開発においては IC を安価で効率的に生産することの重要性が増している．安価で効率的に IC を生産するための方法として，設計や製造工程の一部をサードパーティに委託する手段が用いられている [1]．ハードウェアの設計・製造段階では，設計書の作成に始まり，IP コアの設計やレイアウト設計，製造など様々な段階が存在しており，各段階でサード

1 早稲田大学基幹理工学研究科情報理工・情報通信専攻
Dept. Computer Science and Communications Engineering, Waseda Universityy
2 株式会社 KDDI 総合研究所
KDDI Research, Inc.
a) kazuki.yamashita@togawa.cs.waseda.ac.jp

© 2021 Information Processing Society of Japan

パーティが関与する可能性がある．

ハードウェア製造の過程にサードパーティが関与する中で，悪意のある第三者にハードウェアトロイを挿入されるリスクが報告されている [2], [3]．ハードウェアトロイとは，IC 製品などに挿入された悪意のある動作を起こす回路のことであり，トリガ部とペイロード部によって構成される [4]．トリガ部は論理回路が特定の状態になるまで回路内を監視し，特定の状態になったときにペイロード部を起動する．トリガ部では偶発的に動作が起きないようにするため，極めてまれな信号や，複数の信号の組み合わせが起動条件として設定されることが多い．トリガ回路によって起動状態になったペイロード部は実際に悪意のある動作を実行する．ハードウェアトロイは情報漏洩や機能障害，劣化加速などの思わぬ動作を発生させることで IC 製品に甚大な被害を引き起こすため，システムが製品化される前に検知する必要がある．しかし，ハードウェアトロイは一般的に小規模かつ極めてまれな条件においてのみ起動する回路であるため，検知が困難である．

ハードウェアトロイを検知するために，機械学習を用いた手法の研究が進んでいる．本稿では，特に回路の設計段階における識別手法に焦点を当てる．回路の設計段階は簡単にハードウェアトロイを挿入できるために，攻撃者に狙われやすい [2]．また，回路の設計段階でハードウェアトロイを識別できれば製造段階や製造後にハードウェアトロイを除去する必要がなくなるため，設計段階でハードウェアトロイを識別する重要性は極めて高いといえる．設計情報を用いた識別手法として，ニューラルネットワークを用いる手法 [5] や，ランダムフォレストを用いる手法 [6] が存在する．これらの手法は，回路の設計情報であるネットリストをもとにハードウェアトロイを構成する信号線を識別する．これらの手法はハードウェアトロイ識別において有効な手法であるが，経験則に基づいて特徴量を決定する必要性や，回路の周辺情報を考慮して学習できないなどの欠点も存在する．

本稿では，回路の設計情報を用いてハードウェアトロイを識別する新たな手法として，グラフニューラルネットワーク (GNN; Graph Neural Network) を用いた識別手法を提案する．本手法は回路の設計情報であるネットリストをもとにグラフを作成し，グラフ構造をもとにグラフの各ノードの特徴を学習することでハードウェアトロイを識別する手法である．回路の構成はグラフ構造との親和性が高く，ハードウェアトロイ識別に十分応用可能であると考えられる．実際，RTL を対象に回路にハードウェアトロイが挿入されているかどうかを識別する手法も提案されている [7]．なお文献 [7] は，回路にハードウェアトロイが挿入されているかかどうかを識別するのみで，ハードウェアトロイの挿入箇所の特定は実現されていない．ハードウェアトロイ識別においては，ハードウェアトロイの挿入箇所を特定し，ハードウェアトロイのみを除去することが実用上求められるため，ゲート毎にハードウェアトロイを構成するゲートかどうか判定することが重要である．提案手法はグラフ学習を用いてゲート毎にハードウェアトロイかどうか識別する初めての試みである．

グラフ学習をハードウェアトロイ識別に適用する利点を以下に示す．

- ニューラルネットワークやランダムフォレストによる識別手法では，特徴量が明らかになった場合に攻撃者がその情報を基に識別器に検知されないようなハードウェアトロイを作成する可能性があった．しかし，GNN は回路の接続状況を考慮して特徴量を更新した後に識別するため，識別器に検知されないようなハードウェアトロイを作成することは困難である．
- ニューラルネットワークやランダムフォレストによる識別手法では，周辺の回路素子の接続を考慮した学習が困難であったが，GNN は近傍の回路素子の接続情報を基に学習を進めることができる．
- GNN は周辺のノード情報を含めて特徴量を更新するため，ヒューリスティックに特徴量を選定する必要がない．

グラフ学習の適用は上記に挙げた利点が存在するが，適用する際の課題も存在する．ハードウェアトロイ識別にグラフ学習を使用する際の課題を以下に示す．

- GNN において，各ノードは層数に対応したホップ数に存在するノード情報しか学習できないため，遠方のノード情報を学習できない．
- ハードウェアトロイは通常の回路に比べて極めて小規模であるため，不均衡なデータをうまく学習する必要がある．

本稿で提案する手法では，1 つ目の課題を特徴ベクトルに遠方のノード情報を含めることで，2 つ目の課題をハードウェアトロイの繰り返し学習と損失関数によって解決している．その結果，GNN を用いた場合でも，ヒューリスティックに特徴量を選定することなく，ニューラルネットワークを用いた手法と同等な精度の 79.8% の確率でハードウェアトロイを識別できることを確認した．

本稿は以下のように構成される．2 章で，グラフ学習の一般的な手法について説明した後，本稿で適用した GAT (Graph Attention Networks) と呼ばれるグラフ学習手法について説明する．3 章で，本稿で提案するハードウェアトロイ識別におけるグラフ学習手法について説明する．4 章で提案手法によるハードウェアトロイ識別を実験する．5 章で本稿をまとめる．

2. グラフ学習

本章では，グラフ学習の一般的な手法 [8] と，本稿で適用した GAT と呼ばれるグラフ学習手法 [9] について説明す

る．2.1 節でグラフ学習手法の概要について説明する．2.2 節で GAT による学習手法について説明する．

2.1 GNN

GNN はグラフ領域で表現されるデータを処理するために既存のニューラルネットワーク手法を拡張した手法である [8]．GNN の処理は大きく分けて 2 つのステップから成る．2 つのステップを以下に示す．

(1) AGGREGATE : 更新ノードの近傍にあるノード情報を集約する．

(2) COMBINE : 集約したノードから，更新ノードのノード情報を更新する．

AGGREGATE の処理では，対象とするノードから何段先のノードまでの情報を集約するか決定する必要がある．k 段先までのノード情報を集約する場合，AGGREGATE 関数と COMBINE 関数は以下のように示される [10]．

$$a_v^{(k)} = \mathrm{AGGREGATE}^{(k)}(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\}) \quad (1)$$

$$h_v^{(k)} = \mathrm{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)}) \quad (2)$$

式 (1) では，対象とするノード v に隣接するノード群 u の特徴ベクトル $h_u^{(k-1)}$ を集約し，ベクトル $a_v^{(k)}$ を生成する．式 (2) において，生成された $a_v^{(k)}$ と，$k-1$ 段目までで生成されたノード v の特徴ベクトル $h_v^{(k-1)}$ を用いて新たな特徴ベクトル $h_v^{(k)}$ を生成する．

2.2 GAT

本節では，本稿で提案する手法で用いた GAT と呼ばれるグラフ学習手法を説明する [9]．GAT は更新するノードの近傍にあるノードの重要度を考慮して特徴ベクトルを更新する手法である．

グラフを $G = (V, E)$ とし，G の隣接行列を $A \in \mathbb{R}^{N \times N}$ とする．N はノードの総数である．GAT の入力には隣接行列 A と全ノードの特徴ベクトル $\mathbf{h} = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$ が与えられ，更新された特徴ベクトル $\mathbf{h} = \{\vec{h}'_1, \vec{h}'_2, ..., \vec{h}'_N\}, \vec{h}'_i \in \mathbb{R}^{F'}$ が出力される．F と F' はそれぞれ特徴ベクトルの次元数を示す．

GAT ではまず初めに重み行列 $\mathbf{W} \in \mathbb{R}^{F' \times F}$ を全てのノードに適用したのち，$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ の変換を行い，隣接するノードの重要度を計算する．

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j) \quad (3)$$

式 (3) の計算には単層のニューラルネットワークを用いる．e_{ij} にソフトマックス関数をかけた結果をアテンション係数と呼び，以下の式で計算する．

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \quad (4)$$

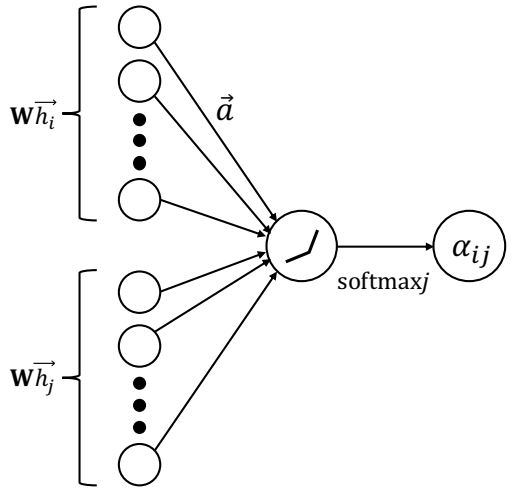

図 1 ノード i におけるノード j の重要度 α_{ij} を単層ニューラルネットワークとソフトマックス関数で求める図．

Fig. 1 Determine the importance α_{ij} of node j in node i with a single layer neural network and softmax function.

また，特徴ベクトル $\vec{h}_i$ と $\vec{h}_j$ を用いてアテンション係数を導出する様子を図 1 に示す．本稿で提案する手法における各ノードの特徴ベクトルは長さ 8 のベクトルであるため，ニューラルネットワークの入力層のユニット数は 16 個となる．活性化関数には LeakyReLU 関数を用いる．α_{ij} と重み行列 W を用いて，以下の式で特徴ベクトルを更新する．

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j\right) \quad (5)$$

式 (5) では，自身を含めた隣接する全てのノードを集約し，それぞれのアテンション係数を計算することで特徴ベクトルを更新する．

GAT では上記の手法を拡張させ，ヘッドと呼ばれる複数の重み行列とアテンション係数の組を用意し，学習結果を安定させるマルチヘッドと呼ばれる手法を用いる．中間層では得られた特徴ベクトルを結合し，出力層では得られた特徴ベクトルの平均を出力とする．K 組のヘッドを用意した場合，中間層の計算は式 (6) で，出力層の計算は式 (7) で表される．

$$\vec{h}'_i = \mathop{\Big\|}_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \quad (6)$$

$$\vec{h}'_i = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \quad (7)$$

式 (6) において，$\|$ はベクトルの結合を表す．そのため，更新された特徴ベクトル $\vec{h}'_i$ は KF' 次元の特徴を持つ．式 (7) では，ベクトルの結合の代わりに K 個の特徴ベクトルの平均を計算することで，最終的な特徴ベクトルを出力する．

3. ハードウェアトロイ識別におけるグラフ学習

本章では，グラフ学習を利用したハードウェアトロイ識別手法を提案する．

3.1 節で提案手法の識別手順について説明する．3.2 節で識別対象に与える特徴ベクトルについて説明する．3.3 節でデータの不均衡さを是正するための損失関数について説明する．3.4 節でハードウェアトロイの繰り返し学習手法について説明する．

3.1 識別手順

本稿で提案する手法の識別手順は以下の通りである．

(1) 回路の設計情報をグラフに変換する．
(2) 各ノードにノードの情報を表す特徴ベクトルを与える．
(3) 特徴ベクトルを保持したグラフを基に学習，識別を実施する．

手順 (1) では，ゲートレベルネットリストにおける各ゲートをノード，信号線をエッジに対応させることでグラフを作成する．手順 (2) では，3.2 節で説明する 8 次元の特徴ベクトルを各ノードに与える．手順 (3) では，各ノードがハードウェアトロイを構成するノードかそうでないかが既知であるグラフを用いて学習を実施する．識別の際には，未知のグラフの各ノードについて，ハードウェアトロイを構成するノードかそうでないかを識別する．以降の説明においてハードウェアトロイを構成するゲートをトロイゲート，通常の回路を構成するゲートをノーマルゲートと呼ぶ．

提案手法の構成を図 2 に示す．提案手法では，学習の際も識別の際も，GAT による特徴ベクトルの更新を 2 回実施し，各ノードがトロイゲートかノーマルゲートかを出力する．学習の際は出力と正解ラベルを損失関数に与え，損失を計算する．本手法では特徴ベクトルの更新を 2 回するため，各ノードは自分から 2 ホップまで離れたノードの情報を集約することとなる．出来上がったモデルを用いて，識別対象である未知のネットリストから作成したグラフの各ノードがトロイゲートかノーマルゲートか識別する．

3.2 特徴ベクトル

グラフ学習ではグラフを作成する際に各ノードに特徴ベクトルを付与する．本節では，提案する特徴ベクトルの付与方法について説明する．特徴ベクトルには回路素子の特徴を表す値を付与する必要があり，図 3 にその一例を示す．図 3 では，論理ゲートとプライマリ入出力をノード，信号線をエッジとして，回路を有向グラフで表現している．図中の各ノードの近くに与えられたベクトルが特徴ベクトルであり，プライマリ入力，プライマリ出力，NOT ゲート，それ以外の回路素子を区別したワンホットなベクトルを付与する．

本手法では回路素子の区別に加えて，ハードウェアトロイ識別の特徴量を考察した先行研究 [6] を参考に，ハードウェアトロイの特徴をより効果的に与える．先行研究 [6] では，ハードウェアトロイはプライマリ入力をトリガーとして使用するためにプライマリ入力の近くに配置されるケースや，プライマリ出力を利用して悪意のある信号を出力するためにプライマリ出力の近くに配置されるケースが多いことが報告されている．また，まれな条件においてのみハードウェアトロイが起動するように設計されるため，ファンイン数の増加や回路素子の種類にも特徴が表れることも報告されている．

以上に示すハードウェアトロイ特有の特徴を基に，本手法では回路素子の種別と共に，プライマリ入出力の距離情報を特徴ベクトルに与える．プライマリ入出力の距離情報を特徴ベクトルに与える手法は，グラフ学習においてノードの位置を考慮する手法である Position-aware 手法 [11] を参考にしている．ハードウェアトロイ識別において重要な指標である，プライマリ入出力の距離情報を用いてノードの位置を表現する本手法は，Position-aware 手法をハードウェアトロイ識別に最適化した手法だといえる．

ファンイン数もハードウェアトロイ識別において重要だが，グラフ学習では各ノードの接続情報を考慮して学習を進めるため，ファンイン数をノードに与える必要はなく，回路素子の特徴を示す情報が必要になると考えられる．提案手法では，回路素子の種別を以下の 5 つに区別する．

- インバータ，バッファ．
- フリップフロップ．
- マルチプレクサ．
- 定数．
- その他の論理ゲート．

回路素子が論理ゲートの場合は，まれな条件においてのみ起動するトリガ部の特徴などを考慮し，論理ゲートの 0/1 出力確率の情報を特徴ベクトルに与える．

以上より，提案手法ではプライマリ入出力の情報と回路素子の種別を合わせた 8 次元の特徴ベクトルを各ノードに与える．図 4 にフリップフロップと 2 入力 AND を例に特徴ベクトルを与えた例を示す．プライマリ入出力の距離情報は標準化した値を与えている．2 入力 AND は論理ゲートであり，4 種類の入力パターンに対して 1 が出力されるパターンが 1 種類のみであるため，0 が出力される確率は 0.75，1 が出力される確率は 0.25 と計算し，図 4 に示したような特徴ベクトルが与える．

3.3 損失関数による重み付け

ハードウェアトロイは通常の回路と比較すると小規模であり，学習の際にはデータ数の不均衡さが発生する．そのため，本稿で提案する手法では重みつきバイナリクロスエ

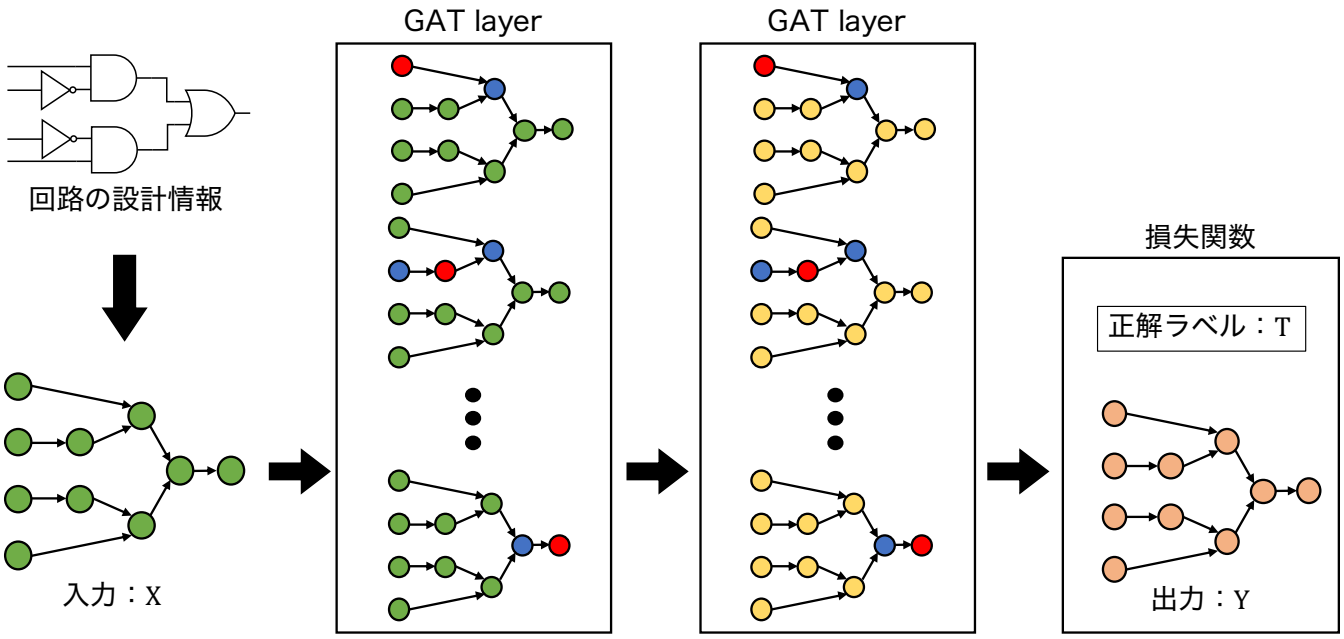

図 2　提案手法の構成.

Fig. 2　Structure of the proposed method.

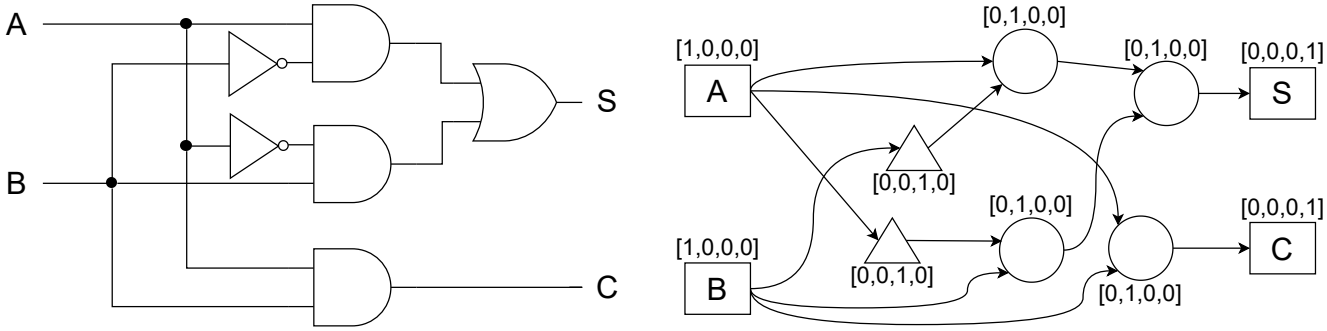

図 3　回路図をグラフ情報に変換し，特徴ベクトルを付与した例.

Fig. 3　An example of converting a circuit diagram into graphical information and adding a feature vector.

ントロピーを使用する．損失関数 E は次のように表せる．

$$E = -\frac{1}{2}\sum_{k=1}^{2}(p_k t_k \log y_k + (1-t_k)\log(1-y_k)) \quad (8)$$

t_k は正解値，y_k は出力値，p_k は重みである．今回説明する手法では，重みベクトル $\boldsymbol{p} = (p_1, p_2)$ をテストデータにおけるトロイゲートとノーマルゲートの総数の比によって決定する．p_1 はノーマルゲートに対応する重み，p_2 はトロイゲートに対応する重みを表す．例えば，ノーマルゲートの総数が 1000 個，トロイゲートの総数が 50 個であった場合，$\boldsymbol{p} = (1, 20)$ となる．

3.4　トロイ回路の繰り返し学習を用いた不均衡データの学習

提案手法では損失関数に加え，ノーマルゲートを分割してミニバッチを作成し，トロイゲートを繰り返し学習することでノーマル/トロイゲート間の学習データ数の不均衡さを是正する．学習データのネットリストがトロイゲート n 個，ノーマルゲート m 個で構成されており，トロイゲートの繰り返し学習回数を k 回に設定した場合，学習データは $a = \mathrm{ceil}(m/k)$ 個のノーマルゲートと n 個のトロイゲートを含んだミニバッチ k 個に分割される．ノーマルゲートを分割する際は，エポック毎にランダムに分割することで異なる情報を学習する．トロイゲートの学習回数はバッチ数と一致するため，合計で a 回学習される．1 つのミニバッチにおける ノーマル : トロイ の比率は $a : n$ となり，この比率によって識別精度が変化する．

繰り返し学習の回数は、TPR (True Positive Rate) と TNR (True Negative Rate) のトレードオフや時間的コストを考慮して適当な値を決めるのが望ましい．今回は学習回数を 1 ～ 50 回の間で変化させ実験し，最適な繰り返し学習回数を決定している（詳細は 4.2.1 項を参照）．

	特徴量ベクトル							
ゲート種別	プライマリ入力までの距離	プライマリ出力までの距離	インバータ，バッファ	フリップフロップ	マルチプレクサ	定数	論理ゲート	
							0出力確率	1出力確率
フリップフロップ	0.3	-0.4	0	1	0	0	0	0
2入力AND	-0.2	0.5	0	0	0	0	0.75	0.25

図 4 提案する特徴ベクトル.
Fig. 4 Proposed feature vector.

4. 評価実験

本章では，3 章で提案した手法をベンチマークを用いて評価した結果を示す．4.1 節で実験条件を説明する．4.2 節で実験結果を示す．4.3 節で実験結果をもとに既存手法と提案手法を比較する．

4.1 実験条件

本稿の実験では Trust-HUB [12], [13], [14] に公開されているベンチマーク回路のうち，17 種類のゲートレベルネットリストを使用した．表 1 に使用したベンチマーク回路と，回路中に含むトロイゲートとノーマルゲートの総数を示す．17 種類のベンチマークのうち，1 種類をテストデータ，残り 16 種類を学習データとする 1 個抜き交差検証を実施した．

モデルの構成として，GAT 層を 2 層，ヘッド数を 6 個とする．入力層のユニット数が 8，GAT の 1 層目のユニット数が 64，出力層となる GAT の 2 層目のユニット数を 2 とした．モデルの最適化アルゴリズムに Adam を使用し，学習率を 0.005，エポック数を 200 とし，活性化関数にはシグモイド関数を使用した．損失関数には，3.3 節で説明したように重みつきバイナリクロスエントロピーを使用し，学習データにおけるノーマルゲートとトロイゲートの比を重みとして与える．また，各層の入力をそのまま次の層に伝える Skip 接続 [15] も適用した．実験には，CPU が Intel Xeon E7-8855 v4，メモリ容量が 1TB のサーバを使用した．

実験で使用した評価指標について説明する．ハードウェアトロイ識別は 2 値分類であるため，ノーマルゲートをノーマルゲートと正しく識別した数 TN (True Negative)，ノーマルゲートをトロイゲートと誤って識別した数 FP (False Positive)，トロイゲートをノーマルゲートと誤って識別した数 FN (False Negative)，トロイゲートをトロイゲートと正しく識別した数 TP (True Positive) を指標に用いる．上記の指標を用いて，TP/(TP + FN) で定義される TPR，TN/(TN+FP) で定義される TNR，TP/(TP+FP) で定義される Precision，2(TPR × Precision)/(TPR + Precision) で定義される F-measure，(TP+TN)/(TP+FN+FP+TN) で定義される Accuracy を評価指標として用いる．

表 1 実験に使用したネットリストのデータ.
Table 1 List of benchmarks utilized in the experiment.

ネットリスト名	ノーマルゲート数	トロイゲート数
RS232-T1000	289	13
RS232-T1100	293	11
RS232-T1200	296	10
RS232-T1300	290	9
RS232-T1400	290	12
RS232-T1500	291	13
RS232-T1600	290	13
s15850-T100	2397	27
s35932-T100	5967	15
s35932-T200	5962	15
s35932-T300	5975	26
s38417-T100	5656	12
s38417-T200	5656	15
s38417-T300	5688	15
s38584-T100	7064	9
s38584-T200	7064	83
s38584-T300	7064	731

4.2 実験結果

本節では，提案手法をベンチマークを用いて評価した結果を示す．4.2.1 項でトロイゲートの繰り返し学習回数を 1 ～ 50 回で変化させたときの識別結果を示す．4.2.2 項で最適なモデルの詳細な識別結果を示す．

4.2.1 繰り返し学習回数の変化による識別結果の比較

トロイゲートの繰り返し学習回数を変化させたときの識別結果を表 2 に示す．表 2 では，各モデルにおける 17 種類ベンチマークの 1 個抜き交差検証の識別結果の平均値を示している．表 2 より，バッチ数の増加に伴い，学習時間が増加していることが分かる．TNR と F-measure 及び Accuracy に関しては繰り返し学習を適用しないバッチ数 1 の場合が最も高く，Precision に関してはバッチ数を 50 にした場合が最も高い．TPR に関してはバッチ数が 5，6，10 のときに良い結果となっており，これは 3.4 節で述べたよ

うに，ミニバッチにおけるノーマルゲートとトロイゲートの比率が最適だったからだと考えられる．ハードウェアトロイ識別の観点においては，ノーマルゲートをトロイゲートと誤識別するよりも，トロイゲートをノーマルゲートと誤識別する危険性の方が高いため TPR の指標を重視するが，3.4 節で述べたように，TNR とのトレードオフや時間的コストなども考慮する必要がある．そのため，TPR の指標が実験結果の中で最も高い 79.80% でありながら TNR を 98% 以上に保ち，学習時間も現実的な時間であるバッチ数 5 が提案手法において最適なバッチ数だと考えられる．

4.2.2 最適なモデルの識別結果

繰り返し学習回数を 5 回としたときの詳細な実験結果を表 3 に示す．表 3 より，TP の値が全てのネットリストにおいて 1 以上となっており，全てのネットリストにおいて少なくとも 1 つ以上のトロイゲートの識別に成功したことが分かる．TPR に関しては 8 つのネットリストにおいて 100% を達成している．TNR に関しては，全てのネットリストで 95% 以上となっており，提案手法においてトロイゲートとノーマルゲートを効果的に識別できたことを確認した．一方で，5 つのネットリストにおいて Precision が 50% を下回っている．これは，トロイゲートの繰り返し学習回数を増加させたことにより，識別器がよりノーマルゲートをトロイゲートと誤認識しやすくなったからだと考えられる．Precision は TPR とトレードオフの関係にあるが，Precision が低すぎるとトロイゲートを適切に取り除くことが困難になるため，TPR の値を維持した上で Precision を高く保つことが今後の課題となる．

4.3 既存手法との比較

本節では，ハードウェアトロイ識別の既存手法であるニューラルネットワークによるハードウェアトロイ識別手法 [5] とランダムフォレストによるハードウェアトロイ識別手法 [6] との 17 種類ベンチマークの識別結果を比較する．比較結果を表 4 に示す．表 4 より，提案手法はニューラルネットワークによる手法より TPR が 5.2 ポイント低く，TNR は 28.9 ポイント高い．ランダムフォレストによる手法については，提案手法は TPR が 5.5 ポイント高く，TNR は 0.8 ポイント低い．ハードウェアトロイ識別においては TPR が最も重要な指標となるが，TNR が低いとハードウェアトロイが挿入されていない回路を誤って廃棄する可能性が高くなるため，両者のバランスを取ることが望ましい．そのため，TPR を 75% 以上に保ちつつ TNR を 98.9% とする本手法は 2 つの既存手法に比べて両者のバランスが取れており，有効な識別手法であると考えられる．

また，2 つの既存手法はハードウェアトロイ特有の特徴量を回路の設計情報から検証し，ヒューリスティックに 51 種類の特徴量を提案した後にランダムフォレスト手法によって抽出した 11 種類の効果的なハードウェアトロイ特徴量を使用している．本稿で提案した手法は回路素子の種別とプライマリ入出力の距離情報のみを特徴量として与えただけにも関わらず既存手法と同等の識別性能となっているため，回路の接続情報を含めて識別を実施する本手法は未知のハードウェアトロイに対する汎用性が期待できる．

5. おわりに

本稿では，ゲートレベルネットリストを対象として，グラフ学習を用いたハードウェアトロイ識別手法の提案と評価を実施した．実験では GAT と呼ばれるグラフ学習手法を用いてハードウェアトロイ識別を実施し，提案手法の有効性を確認できた．今後の課題として，より大規模で複雑な回路のハードウェアトロイ識別の実施など，トロイゲートの識別が困難なゲートレベルネットリストを対象とすることが挙げられる．

参考文献

[1] Rostami, M., Koushanfar, F., Rajendran, J. and Karri, R.: Hardware security: Threat models and metrics, *Proceedings of 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, pp. 819–823 (2013).

[2] Francq, J. and Frick, F.: Introduction to hardware Trojan detection methods, *Proceedings of 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp. 770–775 (2015).

[3] Xiao, K., Forte, D., Jin, Y., Karri, R., Bhunia, S. and Tehranipoor, M.: Hardware trojans: Lessons learned after one decade of research, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol. 22, No. 1, pp. 1–23 (2016).

[4] Bhunia, S., Hsiao, M. S., Banga, M. and Narasimhan, S.: Hardware Trojan Attacks: Threat Analysis and Countermeasures, *Proceedings of the IEEE*, Vol. 102, No. 8, pp. 1229–1247 (2014).

[5] Hasegawa, K., Yanagisawa, M. and Togawa, N.: Empirical Evaluation and Optimization of Hardware-Trojan Classification for Gate-Level Netlists Based on Multi-Layer Neural Networks, *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 101, No. 12, pp. 2320–2326 (2018).

[6] Hasegawa, K., Yanagisawa, M. and Togawa, N.: Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier, *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, pp. 1–4 (2017).

[7] Yasaei, R., Yu, S.-Y. and Al Faruque, M. A.: Gnn4tj: Graph neural networks for hardware trojan detection at register transfer level, *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp. 1504–1509 (2021).

[8] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. and Monfardini, G.: The graph neural network model, *IEEE transactions on neural networks*, Vol. 20, No. 1, pp. 61–80 (2008).

[9] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y.: Graph attention networks, *arXiv preprint arXiv:1710.10903* (2017).

表 2 ハードウェアトロイの繰り返し学習回数による識別性能の変化.

Table 2 Classification results when the iterative trainings are changed.

バッチ数	TPR	TNR	Precision	F-measure	Accuracy	学習時間 [h]
1	74.31%	99.63%	73.97%	70.76%	99.44%	4.26
2	76.80%	99.33%	67.86%	66.81%	99.12%	6.36
3	75.05%	99.37%	66.86%	66.69%	99.16%	8.61
4	77.68%	98.87%	57.59%	63.65%	98.72%	11.2
5	79.80%	98.91%	61.67%	64.32%	98.81%	12.36
6	79.23%	98.79%	56.72%	62.52%	98.65%	13.96
10	78.01%	99.06%	57.93%	62.45%	98.87%	21.63
30	74.35%	98.85%	55.21%	61.36%	98.63%	40.62
50	70.32%	99.58%	82.04%	69.22%	99.27%	66.08

表 3 識別結果.

Table 3 The classification results.

ネットリスト名	TN	FP	FN	TP	TPR	TNR	Precision	F-measure	Accuracy
RS232-T1000	280	9	0	13	100.0%	96.9%	59.1%	74.3%	97.0%
RS232-T1100	284	9	1	10	90.9%	96.9%	52.6%	66.7%	96.7%
RS232-T1200	294	2	0	10	100.0%	99.3%	83.3%	90.9%	99.3%
RS232-T1300	288	2	0	9	100.0%	99.3%	81.8%	90.0%	99.3%
RS232-T1400	285	5	1	11	91.7%	98.3%	68.8%	78.6%	98.0%
RS232-T1500	287	4	0	13	100.0%	98.6%	76.5%	86.7%	98.7%
RS232-T1600	288	2	3	10	76.9%	99.3%	83.3%	80.0%	98.3%
s15850-T100	2335	62	1	26	96.3%	97.4%	29.5%	45.2%	97.4%
s35932-T100	5966	1	0	15	100.0%	100.0%	93.8%	96.8%	100.0%
s35932-T200	5962	0	12	3	20.0%	100.0%	100.0%	33.3%	99.8%
s35932-T300	5974	1	7	19	73.1%	100.0%	95.0%	82.6%	99.9%
s38417-T100	5630	26	6	6	50.0%	99.5%	18.8%	27.3%	99.4%
s38417-T200	5605	51	8	7	46.7%	99.1%	12.1%	19.2%	99.0%
s38417-T300	5632	56	0	15	100.0%	99.0%	21.1%	34.9%	99.0%
s38584-T100	7002	61	8	1	11.1%	99.1%	1.6%	2.8%	99.0%
s38584-T200	7043	21	0	83	100.0%	99.7%	79.8%	88.8%	99.7%
s38584-T300	6994	70	0	731	100.0%	99.0%	91.3%	95.4%	99.1%

表 4 既存手法との比較.

Table 4 Comparison to existing methods.

手法	識別方法	平均 TPR	平均 TNR
[5]	ニューラルネットワーク	85.0%	70.0%
[6]	ランダムフォレスト	74.3%	99.7%
提案手法	グラフ学習	79.8%	98.9%

[10] Xu, K., Hu, W., Leskovec, J. and Jegelka, S.: How powerful are graph neural networks?, *arXiv preprint arXiv:1810.00826* (2018).

[11] You, J., Ying, R. and Leskovec, J.: Position-aware graph neural networks, *International Conference on Machine Learning*, PMLR, pp. 7134–7143 (2019).

[12] : Trust-HUB, `https://www.trust-hub.org/`.

[13] Salmani, H., Tehranipoor, M. and Karri, R.: On design vulnerability analysis and trust benchmarks development, *2013 IEEE 31st international conference on computer design (ICCD)*, IEEE, pp. 471–474 (2013).

[14] Shakya, B., He, T., Salmani, H., Forte, D., Bhunia, S. and Tehranipoor, M.: Benchmarking of hardware trojans and maliciously affected circuits, *Journal of Hardware and Systems Security*, Vol. 1, No. 1, pp. 85–102 (2017).

[15] He, K., Zhang, X., Ren, S. and Sun, J.: Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (2016).